UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/693,589	10/24/2003	Jeffrey P. Snover	MS1-1742US	1106

22801 7590 07/20/2007
LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
SINGH, RACHNA	
ART UNIT	**PAPER NUMBER**
2176	

MAIL DATE	DELIVERY MODE
07/20/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

ED

Office Action Summary	Application No.	Applicant(s)	
	10/693,589	SNOVER ET AL.	
	Examiner	Art Unit	
	Rachna Singh	2176	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 May 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to communications: A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 05/15/07 has been entered.

2. Claims 1-37 are pending. Claims 1, 12, 20, and 26 are independent claims. Claims 1, 12, 20, 26, 33, 35 and 36 have been amended.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to

be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1-37 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 1-37 of copending Application No. 10/882,828

Initially it should be noted that the present application is a parent application of continuation application 10/882,828, having the same inventive entity. The entire disclosures of the instant application and the continuation application are identical.

Although the conflicting claims are not identical, they are not patentably distinct from each other because:

Independent claim 1 of the instant application is almost identical to independent claim 1 of Application no. 10/882,828 except that claim 1 in the instant application recites, ***the parseable object includes at least one method***; whereas, claim 1 in Application No. 10/882,828 recites that ***the parseable object having no methods.*** The limitations above describe different variations of the parseable object (i.e. has a method or does not have a method). The parseable object having a method or not having a method makes no difference in the outcome of the claim and is not sufficient to render the claim patentably distinct and therefore a terminal disclaimer is required. Thus claim 1 merely recites an obvious variation of the invention claimed in US Application No. 10/882,828 of the object.

Claims 2-11 in the instant application are identical to claims 2-11 of application no. 10/882,828.

Independent claim 12 of the instant application is almost identical to independent claim 12 of Application no. 10/882,828 except that claim 12 in the instant application recites, ***the parseable object includes at least one method***; whereas, claim 12 in Application No. 10/882,828 recites that ***the parseable object having no methods.*** The limitations above describe different variations of the parseable object (i.e. has a method or does not have a method). The parseable object having a method or not having a method makes no difference in the outcome of the claim and is not sufficient to render the claim patentably distinct and therefore a terminal disclaimer is required. Thus claim 12 merely recites an obvious variation of the invention claimed in US Application No. 10/882,828 of the object.

Claims 13-19 in the instant application are identical to claims 13-19 of application no. 10/882,828.

Independent claim 20 of the instant application is almost identical to independent claim 20 of Application no. 10/882,828 except that claim 20 in the instant application recites, ***the parseable object includes at least one method***; whereas, claim 20 in Application No. 10/882,828 recites that ***the parseable object having no methods.*** The limitations above describe different variations of the parseable object (i.e. has a method or does not have a method). The parseable object having a method or not having a method makes no difference in the outcome of the claim and is not sufficient to render the claim patentably distinct and therefore a terminal disclaimer is required.

Thus claim 20 merely recites an obvious variation of the invention claimed in US Application No. 10/882,828 of the object.

Claims 21-25 in the instant application are identical to claims 21-25 of application no. 10/882,828.

Independent claim 26 of the instant application is almost identical to independent claim 26 of Application no. 10/882,828 except that claim 26 in the instant application recites, ***the parseable object includes at least one method***; whereas, claim 26 in Application No. 10/882,828 recites that ***the parseable object having no methods.*** The limitations above describe different variations of the parseable object (i.e. has a method or does not have a method). The parseable object having a method or not having a method makes no difference in the outcome of the claim and is not sufficient to render the claim patentably distinct and therefore a terminal disclaimer is required. Thus claim 26 merely recites an obvious variation of the invention claimed in US Application No. 10/882,828 of the object.

Claims 27-37 in the instant application are identical to claims 27-37 of application no. 10/882,828

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 1-2, 7-13, 15-19, and 20-25 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter for failure to produce a tangible result.

Specifically, independent claims 1, 12, and 20 fail to produce a tangible result. Merely emitting a format object for access by another command does not produce a useful and tangible result since the result is not made available to the user.

It is noted that the dependent claims 3-6 and 14 render the results of the pipeline which is considered statutory because the result is made available to the user. Dependent claims 8-10, 16-18, and 24-25 would be considered statutory if the conversion is made available to the user.

Claims 2, 8-11, 13, 15-19, and 21-25 are rejected under 35 U.S.C. 101 for fully incorporating the deficiencies of their base claim from which they depend.

Independent claim 12 is further rejected under 35 U.S.C. 101 for reciting *a computer-readable medium including at least one tangible component*. Claim 12 is not limited to only tangible embodiments. In view of Applicant's disclosure, specification pages 6-7, the medium is not limited to tangible embodiments, instead being defined as including both tangible embodiments (e.g., [storage media, memory]) and intangible embodiments (e.g., [signal, carrier wave]). As such, the claim is not limited to statutory subject matter and is therefore non-statutory. The claim should be amended to recite only tangible embodiments.

Claims 13-19 are rejected under 35 U.S.C. 101 for fully incorporating the deficiencies of their base claim from which they depend.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

8. Claims 1-37 are rejected under 35 U.S.C. 102(e) as being anticipated by Muhlestein et al., US 2003/0018765 A1 (field 01/23/03).

Regarding claim 1, Muhlestein dislcoses a system and method for accessing management functionality through a command line utility. Muhlestein discloses a set of commands for the WMI command utility configured by an underlying object model command schema. The object-oriented command schema defines the command line utility comprising a plurality of commands which meets the limitation ***a pipeline***

comprising a plurality of object-based commands. See pages 1 and 2, paragraphs [0011]-[0013].

The command line interface permits the entry of command and control functions that are based on and operate against a target WMI schema exposed through the WMI infrastructure which represents the systems, applications, networks, and other managed components of a target system using an alias object. The command line utility executes an alias object which is a command in order to facilitate a specific administrative task (i.e. method or process). The command schema drives the WMI command line utility and defines the commands used in the utility. An example method for implementing the WMI command utility begins when a command is entered into a command line and actually received by an executable file within the WMI command line utility. The utility, through the executable file, performs a series of operations on the command. The utility interprets the command based on the definition and executes the command as a series of WMI API calls where the WMI data (parseable object) retrieved through API calls is transformed into XML information that is readable by the WMI command line utility which meets the limitations, ***receiving a parseable object emitted from a prior object-based command (within a pipeline comprising a plurality of object-based commands) the prior object-based command being one of the plurality of object-based commands*** and ***wherein the parseable object includes at least one method***. The WMI data is returned in XML to the command line utility. See pages 5 and 9.

Muhlestein system *supports* a pipeline of multiple commands. The command line utility executes an alias object which is a command in order to facilitate a specific

administrative task (i.e. method or process). The WMI input is provided to a command that outputs the results which meets the limitation, ***operating environment that supports the pipeline of a plurality of object-based commands is configured to support execution of object based commands within the same process***.

The command line utility allows a user to tailor commands. Properties of a command can be arranged in named formats that include property values that can be formatted for display according to a specific presentation strategy which meets the limitation, ***obtaining a data type for the parseable object***. See pages 5 and 9-10. The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias which meets the limitation, ***obtaining format information describing a format for the data type***. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user which meets the limitations, ***such that a subsequent object-based command within the pipeline which receives the parseable object is configured to communicate with the prior object-based command within the pipeline through the parseable object emitted from the prior object-based command*** and ***emitting a format object for access by another subsequent object-based command, the format object being based on the format information***. See pages 5 and 9-10.

Examiner note: The subsequent object-based command can be an output command configured to render the results of the pipeline based on the received parseable object

and format object. In Muhlestein, the WMI data (parseable object) and format information is used to present the WMI information.

In reference to claim 2, Muhlestein teaches the utility interprets the command based on the definition and executes the command as a series of WMI API calls where the WMI data (parseable object) retrieved through API calls is transformed into XML information that is readable by the WMI command line utility. See page 10, paragraph [0095].

In reference to claim 3, Muhlestein teaches the WMI data (parseable object) and format information is used to present the WMI information. See pages 5 and 9-10. The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

In reference to claim 7, Muhlestein teaches the command schema comprises an alias class defining a command template and a format class as a subclass to the alias class, each instance of the format class representing a list of properties that will be returned through processing an alias. The alias object is a command that is executed in order to capture the features of the target class and to facilitate a task. Alias objects are

instances of command-related classes organized into a command schema. The command schema defines the commands and the command line utility uses the aliases to interpret command information and apply the interpretation against the target schema. Properties of the alias object can be arranged in named formats that include property values that can be formatted for display. See page 5. The formats attribute includes a list of properties that are to be displayed using the given format. See page 34 and page 6, paragraphs [0054]-[0059]. Muhlestein further teaches permitting the generation of additional commands to be added to one or more commands. The user can parameterize commands by specifying locations of command definitions and target system objects. A connection class, a subclass to the alias class, defines the connection instances, each connection instance representing connection parameters used by an alias to establish a connection to target namespace within the target schema. The format class represents a list of properties that will be returned through processing an alias. The command schema is extensible to permit the generation of additional commands to be added to the set of commands. See page 8 and page 34.

Regarding claim 8, The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

Regarding claim 9, Muhlestein teaches the WMI data is transformed into XML information. The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

Regarding claim 10, Muhlestein teaches the WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

In reference to claim 11, Muhlestein teaches properties of the alias object can be arranged in named formats that include property values that can be formatted for display. See page 5. The formats attribute includes a list of properties that are to be displayed using the given format. See page 34 and page 6, paragraphs [0054]-[0059].

Regarding **claims 12-19**, claims 12-29 are drawn to a computer readable medium comprising the instructions for executing the method steps of claim 1. Thus claims 12-19 are rejected under the same rationale used in claims 1-3 and 7-11 respectively above.

Regarding **claims 20-25**, claims 20-25 are drawn to a system comprising the hardware to carrying out the steps of claim 1. Thus claims 20-25 are rejected under the same rationale used in claims 1-2, 11, 7-8, and 10 respectively above.

In reference to claims 26, Muhlestein teaches a command line utility comprising an object model command schema to define a mapping between one or more commands an object model target schema, the one or more commands generated by the command schema and configured to operate against the target schema through the command line. Compare to ***"a method for providing a data driven command line output"***. The command line utility comprises an alias class defining a command template, each alias of the alias class representing a single command and a format class as a subclass to the alias class, each instance of the format class representing a list of properties that will be returned through processing an alias. The command schema defines the commands and the command line utility uses the aliases to interpret command information and apply the interpretation against the target schema. See page 2, page 6, paragraphs [0054]-[0059], and page 34. For example, a command is received through an interface by the WMI command line utility. The utility interprets the command based on its definition. The command is executed as a series of API calls against a target namespace. The WMI infrastructure at the target station then performs the WMI operations against the target object. The WMI data retrieved is transformed at operation into XML information that is readable by the command line utility. See page

9, paragraph [0091]-page 10, paragraph [0095]. Compare to ***"receiving command-line instruction containing an output command configured to receive at least one object, the object having at least one method; and executing the output command to manipulate at least one object"***.

The command line interface permits the entry of command and control functions that are based on and operate against a target WMI schema exposed through the WMI infrastructure which represents the systems, applications, networks, and other managed components of a target system using an alias object. The command line utility executes an alias object which is a command in order to facilitate a specific administrative task (i.e. method or process). The command schema drives the WMI command line utility and defines the commands used in the utility. An example method for implementing the WMI command utility begins when a command is entered into a command line and actually received by an executable file within the WMI command line utility. The utility, through the executable file, performs a series of operations on the command. The utility interprets the command based on the definition and executes the command as a series of WMI API calls where the WMI data (parseable object) retrieved through API calls is transformed into XML information that is readable by the WMI command line utility which meets the limitations, ***wherein the receiving occurs as part of a pipeline of a plurality of object-based commands*** and ***wherein the parseable object includes at least one method***. The WMI data is returned in XML to the command line utility. See pages 5 and 9. The command line utility allows a user to tailor commands. Properties of a command can be arranged in named formats that include property values that can

be formatted for display according to a specific presentation strategy. See pages 5 and 9-10. The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user which meets the limitation, ***such that a subsequent object-based command within the pipeline which receives the parseable object is configured to communicate with the prior object-based command within the pipeline through the parseable object emitted from the prior object-based command***. See pages 5 and 9-10.

Properties of the alias object can be arranged in named formats that include property values that can be formatted for display. See page 5. The formats attribute includes a list of properties that are to be displayed using the given format. See page 34 and page 6, paragraphs [0054]-[0059]. Compare to ***"output a result to an output destination"***.

In reference to claims 27-28, Muhlestein teaches an object-based command-line environment. See abstract and pages 1-2.

In reference to claims 32, Muhlestein teaches the formats attribute includes a list of properties that are to be displayed using the given format. See page 34 and page 6, paragraphs [0054]-[0059]. Muhlestein further teaches the Format string provides a

location of an XSL file that can be used to format the display for the list of properties being returned to the user through the alias. The WMI command line utility returns information as XML documents which are processed using XSL to render reports for the user in multiple formats as determined by the XSL file located through the format string. Any display strategies to be used in displaying properties will be defined by the XSL style sheet in formatting displays. See page 6, paragraph [0063].

Regarding claim 36, the WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

Regarding claim 37, Muhlestein teaches the WMI data is transformed into XML information. The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

Regarding claim 33, Muhlestein teaches the WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user.

The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

Regarding claim 34, The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user.

In reference to claim 35, Muhlestein further teaches permitting the generation of additional commands to be added to one or more commands. The user can parameterize commands by specifying locations of command definitions and target system objects. A connection class, a subclass to the alias class, defines the connection instances, each connection instance representing connection parameters used by an alias to establish a connection to target namespace within the target schema. The format class represents a list of properties that will be returned through processing an alias. The command schema is extensible to permit the generation of additional commands to be added to the set of commands. See page 8 and page 34

In reference to claims 4-6 and 29-31, Muhlestein teaches the formats attribute includes a list of properties that are to be displayed using the given format. See page

34 and page 6, paragraphs [0054]-[0059]. Muhlestein further teaches the Format string provides a location of an XSL file that can be used to format the display for the list of properties being returned to the user through the alias. The WMI command line utility returns information as XML documents which are processed using XSL to render reports for the user in multiple formats as determined by the XSL file located through the format string. Any display strategies to be used in displaying properties will be defined by the XSL style sheet in formatting displays. See page 6, paragraph [0063]. Muhlestein further discloses displaying the formatted data through a graphical user interface. See page 36, claims 34-36 and figures 2 (illustrates a console) and figure 3.

Response to Arguments

9. Applicant's arguments filed 05/15/07 have been fully considered but they are not persuasive.

Claims 1, 12, 20, 26, 33, 35, and 36 have been amended. Claims 1-37 are currently pending.

Regarding claims 1, 12, 20, and 26, Applicant argues on pages 20-21, Muhlestein does not show or disclose *"receiving a parseable object emitted from a prior object-based command within a pipeline comprising a plurality of object-based commands, the prior object-based command being one of the plurality of object-based commands, such that a subsequent object-based command within the pipeline which receives the parseable object is configured to communicate with the prior object-based*

command within the pipeline through the parseable object emitted from the prior object-based command". Applicant states the interpretation of the Muhlestein reference is incorrect. Applicant argues the "set of commands" referred to in Muhlestein simply describes that the WMI command line utility enables administrators to organize class instances into namespaces. Examiner disagrees and maintains that the "set of commands in the command line" disclosed by Muhlestein is a "pipeline comprising a plurality of object-based commands". Muhlestein teaches a command line in which *a set of commands* for a WMI command line utility is entered on the command line. See page 2, paragraph [0012] where Muhlestein states "a set of commands for the WMI command line utility is configured by an underlying object model command schema that defines a mapping between the commands and the WMI schema".

Applicant argues that even if the set of commands were a pipeline, Muhlestein does not mention pipelining commands in the same manner as claim 1. Specifically, Applicant argues Muhlestein does not teach that *a subsequent object-based command within the pipeline which receives the parseable object is configured to communicate with prior object-based command within the pipeline through the parseable object emitted from the prior object-based command.* Examiner disagrees. A parseable object is defined as input in which properties and values may be discerned. One type of parseable object may be Windows Management Instrumentation input (see Applicant's specification, page 19, lines 5-14. In Muhlestein, the parseable object is the WMI input. Muhlestein teaches the command line interface permits the entry of command and control functions that are based on and operate against a target WMI schema exposed

through the WMI infrastructure which represents the systems, applications, networks, and other managed components of a target system using an alias object. The command line utility executes an alias object which is a command in order to facilitate a specific administrative task (i.e. method or process). The command schema drives the WMI command line utility and defines the commands used in the utility. An example method for implementing the WMI command utility begins when a command is entered into a command line and actually received by an executable file within the WMI command line utility. The utility, through the executable file, performs a series of operations on the command. The utility interprets the command based on the definition and executes the command as a series of WMI API calls where the WMI data (parseable object) retrieved through API calls is transformed into XML information that is readable by the WMI command line utility which meets the limitations, ***receiving a parseable object emitted from a prior object-based command (within a pipeline comprising a plurality of object-based commands) the prior object-based command being one of the plurality of object-based commands*** and ***wherein the parseable object includes at least one method***. The WMI data is returned in XML to the command line utility. See pages 5 and 9. The command line utility allows a user to tailor commands. Properties of a command can be arranged in named formats that include property values that can be formatted for display according to a specific presentation strategy which meets the limitation, ***obtaining a data type for the parseable object***. See pages 5 and 9-10. The WMI command line receives the WMI XML data and applies an XSL style sheet to format the text for display to the user. The

XSL style sheet used is based on the XSL file designated through format specifications within the alias which meets the limitation, ***obtaining format information describing a format for the data type***. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user which meets the limitations, ***such that a subsequent object-based command within the pipeline which receives the parseable object is configured to communicate with the prior object-based command within the pipeline through the parseable object emitted from the prior object-based command*** and ***emitting a format object for access by another subsequent object-based command, the format object being based on the format information***. See pages 5 and 9-10.

Applicant's disclosure indicates the subsequent object-based command can be an output command configured to render the results of the pipeline based on the received parseable object and format object. In Muhlestein, the WMI data (parseable object) and format information is used to present the WMI information.

Applicant argues the command line utility in Muhlestein provides a command line and only allows one command to execute a series of API calls and fails to teach an ***operating environment that supports the pipeline of a plurality of object-based commands is configured to support execution of object based commands within the same process***. Examiner disagrees. Muhlestein *supports* a pipeline of multiple commands. The command line utility executes an alias object which is a command in order to facilitate a specific administrative task (i.e. method or process).

Applicant argues the parseable object does not have at least one method in the Muhlestein reference. Examiner disagrees. The command line utility executes an alias object which is a command in order to facilitate a specific administrative task (i.e. method or process).

Applicant argues Muhlestein fails to teach emitting an object for access by a subsequent command and emitting a format object. Examiner disagrees. As stated above, an example method for implementing the WMI command utility begins when a command is entered into a command line and actually received by an executable file within the WMI command line utility. The utility, through the executable file, performs a series of operations on the command. The utility interprets the command based on the definition and executes the command as a series of WMI API calls where the WMI data (parseable object) retrieved through API calls is transformed into XML information that is readable by the WMI command line utility which meets the limitations, ***receiving a parseable object emitted from a prior object-based command (within a pipeline comprising a plurality of object-based commands) the prior object-based command being one of the plurality of object-based commands*** and ***wherein the parseable object includes at least one method***. The WMI data is returned in XML to the command line utility. See pages 5 and 9. The command line utility allows a user to tailor commands. Properties of a command can be arranged in named formats that include property values that can be formatted for display according to a specific presentation strategy which meets the limitation, ***obtaining a data type for the parseable object***. See pages 5 and 9-10. The WMI command line receives the WMI

XML data and applies an XSL style sheet to format the text for display to the user. The XSL style sheet used is based on the XSL file designated through format specifications within the alias which meets the limitation, ***obtaining format information describing a format for the data type***. See pages 5 and 9-10. The XSL style sheet is applied to format the display and present the WMI information for display to a user which meets the limitations, ***such that a subsequent object-based command within the pipeline which receives the parseable object is configured to communicate with the prior object-based command within the pipeline through the parseable object emitted from the prior object-based command*** and ***emitting a format object for access by another subsequent object-based command, the format object being based on the format information***. See pages 5 and 9-10.

Applicant argues the dependent claims are allowable by virtue of their dependency upon an independent claim. The rejections of the dependent claims have been maintained in light of the remarks with respect to the independent claims above.

In view of the comments above, the rejections are maintained.

Conclusion

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Rachna Singh whose telephone number is 571-272-4099. The examiner can normally be reached on M-F (8:30AM-6:00PM).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Doug Hutton can be reached on 571-272-4137. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Rachna Singh
Art Unit 2176